Building a Page Flow Web App

Table of contents

1. Introduction

Now that the theory of Java Page Flows has been explained, you need to know how to concretely put together a web-app that uses JPFs. Beehive ships with a set of Ant buildfiles to make the building of an application much easier.

2. Source Tree Layout

The layout of your project may be anywhere on your local disk. We will assume that the environment variable $WEBAPP_DIR points to the top-level of your application.

Here is a typical directory structure, an emandation of the core directory structure found in the web app template `<BeehiveRoot>/samples/netui-blank`. For instructions on using the web app template, see Project: Page Flow (../pageflow/sample_netui-blank.html) .

- `$WEBAPP_DIR/`
  - `Controller.java`
  - `index.jsp`
  - `login.jsp`
  - `signup.jsp`
  - `mypage.jsp`
  - `resources/`
  - `WEB-INF/`
    - `classes/`
    - `lib/`
    - `src/`

2.1. $WEBAPP_DIR/

The top-level of the web-app, at `$WEBAPP_DIR` should contain your JSP pages and a `Controller.java`. When built, the `Controller.java` will be compiled to `WEB-INF/classes/Controller.class`.

2.2. $WEBAPP_DIR/WEB-INF/

The `$WEBAPP_DIR/WEB-INF/` directory is just as it is with any other servlet-based application.

2.3. $WEBAPP_DIR/WEB-INF/src/

The `$WEBAPP_DIR/WEB-INF/src/` directory contains any other application source files

that need to be compiled into Java classes. They will be compiled to `$WEBAPP_DIR/WEB-INF/classes/`. Additionally, any extra `.properties` or `.xml` files that need to be deployed with your application will be copied from the `src/` directory to the `classes/` directory during the build.

`$WEBAPP_DIR/WEB-INF/src/` should also contain the build related files for the web app. See the web app template `<BeehiveRoot>/samples/netui-blank/WEB-INF/src` for an example build file and supporting `build.properties` file.

2.4. $WEBAPP_DIR/WEB-INF/lib/

As with any other web application, the `$WEBAPP_DIR/WEB-INF/lib/` directory should contain the jars for each dependency of your application, including those required by Beehive itself. The jars required by Beehive are listed below.

Note: this Ant command will deploy the necessary JARs listed below (plus a few other optional control JARs) to the WEB-INF/lib directory.

```
ant -f $BEEHIVE_HOME\ant\beehive-runtime.xml -Dwebapp.dir=$WEBAPP_DIR
deploy.beehive.webapp.runtime
```

Project	Jar	Version
XMLBeans	apache-xbean.jar	2.0.0
Axis	axis.jar	1.2RC2
Axis	axis-ant.jar	1.2RC2
Beehive NetUI	beehive-netui-pageflow.jar	*distribution*
Beehive NetUI	beehive-netuid-scoping.jar	*distribution*
Beehive NetUI	beehive-netui-tags-databinding.jar	*distribution*
Beehive NetUI	beehive-netui-tags-html.jar	*distribution*
Beehive NetUI	beehive-netui-tags-template.jar	*distribution*
Beehive NetUI	beehive-netui-util.jar	*distribution*
Jakarta Commons Bean Utils	commons-beanutils	1.6
Jakarta Commons Codec	commons-codec-1.3.jar	1.3
Jakarta Commons Collections	commons-collections.jar	2.1.1
Jakarta Commons Digester	commons-digester.jar	1.5

Jakarta Commons Discovery	commons-discovery-0.2.jar	0.2
Jakarta Commons EL	commons-el.jar	1.0
Jakarta Commons Servlet File Upload	commons-fileupload.jar	1.0
Jakarta Commons Logging	commons-logging.jar	1.0.3
Jakarta Commons Validator	commons-validator.jar	1.1.3
Beehive Controls	beehive-controls.jar	*distribution*
Jakarta Commons ORO (Text Processing)	jakarta-oro.jar	2.0.8
JAX RPC	jaxrpc.jar	1.1
JSR 173 (Streaming API for XML)	jsr173_1.0_api.jar	1.0
JSTL	jstl.jar	1.1.0-D13
Log4J	log4j-1.2.8.jar	1.2.8
SAAJ	saaj.jar	1.2
JSTL	standard.jar	1.1.0-D13
Struts	struts.jar	1.2.4
Velocity	velocity-1.4.jar	1.4
Velocity	velocity-dep-1.4.jar	1.4
WSDL4J	wsdl4j.jar	1.5
Beehive Web Services	beehive-wsdltypes.jar	*distribution*
Beehive Web Services	beehive-wsm-axis.jar	*distribution*
Beehive Web Services	beehive-wsm.jar	*distribution*

3. Running Ant

The following section assumes that you are using the Ant build.xml file found at `<BeehiveRoot>/samples/netui-blank/WEB-INF/src/`. We assume either (1) that the build.xml file has been copied to `$WEBAPP_DIR/WEB-INF/src/` or (2) that your web app has been developed by amending the template web app

<BeehiveRoot/samples/netui-blank. (For instructions on starting a web with the template web app, see Project: Page Flow (../pageflow/sample_netui-blank.html) .)

You also set the following properties in a supporting `build.properties` file located at `$WEBAPP_DIR/WEB-INF/src/build.properties`.

- `beehive.home` -- points to the top-level directory of your Beehive installation
- `servlet-api.jar` -- for Tomcat, this value is $CATALINA_HOME/common/lib/servlet-api.jar
- jsp-api.jar -- for Tomcat, this value is $CATALINA_HOME/common/lib/jsp-api.jar
- context.path -- determines (1) part of the URL where your web app resides, e.g., http://some.domain/**contextPath**/someDirectory, and (2) the name of the compiled WAR file.

An example `build.properties` file appears below.

build.properties

```
beehive.home=C:/apache/apache-beehive-1.0
servlet-api.jar=${os.CATALINA_HOME}/common/lib/servlet-api.jar
jsp-api.jar=${os.CATALINA_HOME}/common/lib/jsp-api.jar
context.path=contextPath
```

Before you can build the web-app using Ant, you must ensure that the following variables are set: `ANT_HOME`, `JAVA_HOME`, and `CATALINA_HOME`.

Variable	Value
ANT-HOME	Top level of your Ant distribution
JAVA_HOME	You must have JDK5 installed.
CATALINA_HOME	Top level of the installed Tomcat server.

Once these variables are set correctly, building the deployable web-app requires a single ant invocation using the build file at `$WEBAPP_DIR/WEB-INF/src/build.xml`. This build-file is invoked using ant's `-f <buildfile-path>` option. The directory of the web app is gleened from the supporting `build.properties file`. Finally, the `clean`, `build`, and `war` targets are invoked in turn.

```
ant -f $WEBAPP_DIR/WEB-INF/src/build.xml clean build war
```

The `clean` target deletes any old build artifacts from the `WEB-INF/classes` directory, and other directories.

The `build` target compiles new build artifacts from the source files.

Finally, the `war` target zips up the results into a WAR file, named `contextPath.war`.

4. Deploying the Web-app

The easiest way to deploy the web-app is to create a symlink/shortcut from $WEBAPP_DIR to $CATALINA_HOME/webapps.

```
ln -s $WEBAPP_DIR $CATALINA_HOME/webapps
```

Another way would be to simply copy $WEBAPP_DIR to $CATALINE_HOME/webapps.

```
cp -R $WEBAPP_DIR $CATALINE_HOME/webapps
```

Another way is to visit the following link in a browser. This method assumes that you have created the manager role in the configuration file `CATALINA_HOME/conf/tomcat-users.xml`. For details, see Installation and Setup (../setup.html) . <**Context-Path**> is the desired URL path to the application. <**Full-Path-to-Development-Dir**> is the location of the application on your machine.

```
http://localhost:8080/manager/deploy?path=<Context-Path>&war=<Full-Path-to-Development-
```

5. Next...

Now that you've built and deployed an application, you can see how easy it is to modify the flow between pages, adding and removing pages.

- Altering a Page Flow (pageflow_altering.html)

Java, J2EE, and JCP are trademarks or registered trademarks of Sun Microsystems, Inc. in the United States and other countries.